Fractional Pebbling and Thrifty Branching Programs

LIPIcs Leibniz International Proceedings in Informatics

Fractional Pebbling and Thrifty Branching Programs

Mark Braverman[1], Stephen Cook[2], Pierre McKenzie[3], Rahul Santhanam[4], Dustin Wehr[2]

[1] Microsoft Research
Cambridge, Massachussetts
mbraverm@cs.toronto.edu

[2] University of Toronto
Toronto
sacook@cs.toronto.edu, wehr@cs.toronto.edu

[3] Universite de Montreal
Montreal
mckenzie@iro.umontreal.ca

[4] University of Edinburgh
Edinburgh
rsanthan@inf.ed.ac.uk

ABSTRACT.
We study the branching program complexity of the *tree evaluation problem*, introduced in [3] as a candidate for separating **NL** from **LogCFL**. The input to the problem is a rooted, balanced d-ary tree of height h, whose internal nodes are labelled with d-ary functions on $[k] = \{1, \ldots, k\}$, and whose leaves are labelled with elements of $[k]$. Each node obtains a value in $[k]$ equal to its d-ary function applied to the values of its d children. The output is the value of the root.
Deterministic k-way branching programs as related to black pebbling algorithms have been studied in [3]. Here we introduce the notion of *fractional pebbling* of graphs to study non-deterministic branching program size. We prove that this yields non-deterministic branching programs with $\Theta(k^{h/2+1})$ states solving the Boolean problem "determine whether the root has value 1" for binary trees - this is asymptotically better than the branching program size corresponding to black-white pebbling. We prove upper and lower bounds on the fractional pebbling number of d-ary trees, as well as a general result relating the fractional pebbling number of a graph to the black-white pebbling number.
We introduce a simple semantic restriction called *thrifty* on k-way branching programs solving tree evaluation problems and show that the branching program size bound of $\Theta(k^h)$ is tight (up to a constant factor) for all $h \geq 2$ for deterministic thrifty programs. We show that the non-deterministic branching programs that correspond to fractional pebbling are thrifty as well, and that the bound of $\Theta(k^{h/2+1})$ is tight for non-deterministic thrifty programs for $h = 2, 3, 4$. We hypothesise that thrifty branching programs are optimal among k-way branching programs solving the tree evaluation problem - proving this for deterministic programs would separate **L** from **LogCFL** and proving it for non-deterministic programs would separate **NL** from **LogCFL**.

Foundations of Software Technology and Theoretical Computer Science (Kanpur) 2009.
Editors: Ravi Kannan and K. Narayan Kumar; pp 109–120
Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany.
Digital Object Identifier: 10.4230/LIPIcs.FSTTCS.2009.2311

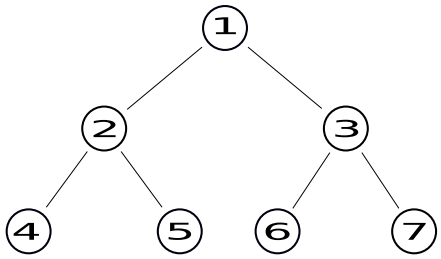

Figure 1: A height 3 binary tree T_2^3 with nodes numbered heap style.

1 Introduction

One of the fundamental problems in complexity theory is to separate **P** from **L**. In a recent paper [3], we propose the Tree evaluation problem as a candidate to separate these classes, and indeed to obtain the much tighter separation of **LogCFL** from **NL**.

The *Tree Evaluation problem* $FT_d(h,k)$ is defined as follows. The input to $FT_d(h,k)$ is a balanced d-ary tree of height h, denoted T_d^h.(see Fig. 1). Attached to each internal node i of the tree is some explicit function $f_i : [k]^d \to [k]$ specified as k^d integers in $[k] = \{1,\ldots,k\}$. Attached to each leaf is a number in $[k]$. Each internal tree node thus takes a value in $[k]$ by applying its attached function to the values of its children. The function problem $FT_d(h,k)$ is to compute the value of the root, and the Boolean problem $BT_d(h,k)$ is to determine whether this value is 1.

In [3], we show that $BT_d(h,k) \in$ **LogCFL**. To show that **LogCFL** $\not\subseteq$ **L** (resp. **LogCFL** $\not\subseteq$ **NL**), it's sufficient to get a super-polynomial lower bound on the deterministic (resp. non-deterministic) branching program complexity of the Tree evaluation problem. As we observe in [3], a lower bound of $\Omega(k^{r(h)})$ on the number of states in any deterministic (resp. non-deterministic) branching program solving $FT_d(h,k)$ or $BT_d(h,k)$ for *any* unbounded function $r(h)$ would yield the desired separation between **LogCFL** and **L** (resp. **NL**).

In this paper, we study the deterministic and non-deterministic branching program complexity of the tree evaluation problem, both from the perspective of upper bounds and lower bounds. In the context of branching programs we think of d and h as fixed, and we are interested in how the number of states required grows with k. To indicate this point of view we write the function problem $FT_d(h,k)$ as $FT_d^h(k)$ and the Boolean problem $BT_d(h,k)$ as $BT_d^h(k)$. For this it turns out that k-way branching programs are a more natural model than Boolean branching programs, since an input of $FT_d^h(k)$ or $BT_d^h(k)$ is naturally presented as a tuple of elements in $[k]$. Each non-final state in a k-way BP queries a specific element of the tuple, and branches k possible ways according to the k possible answers. Lower bounds for k-way BPs are at least as strong as lower bounds for Boolean BPs, while upper bounds can be smaller by at most a factor of k.

Our best upper bounds for k-way deterministic branching programs come from black pebbling algorithms for trees. There is a well-known generalisation of black pebbling called black-white pebbling which naturally models non-deterministic procedures. However we find we can often do better in terms of non-deterministic branching program complexity

than using black-white pebbling. For example, there is a k-way non-deterministic branching program of size $O(k^{5/2})$ which solves $BT_2^3(k)$ while the size of the branching program arising from the optimal black-white pebbling of T_2^3 is $O(k^3)$. These non-trivial upper bounds lead us to re-examine the notion of pebbling, and we come up with a more relaxed notion of pebbling called *fractional pebbling*, which corresponds to non-deterministic branching program complexity in a tighter way. For example, the tree T_2^3 can be fractionally pebbled with $5/2$ pebbles, which leads to non-deterministic branching programs of size $O(k^{5/2})$ for $BT_2^3(k)$. We show a general correspondence between fractional pebbling number and non-deterministic branching program complexity.

THEOREM 1. *If T_d^h can be fractionally pebbled with p pebbles, then non-deterministic branching programs can solve $BT_d^h(k)$ with $O(k^p)$ states*

We explore this new notion of pebbling, and prove a general result that fractional pebbling saves at most a factor of 2 over black-white pebbling. Getting tight bounds on the fractional pebbling number of trees turns out to be much more difficult than proving bounds for black-white pebbling. We do have some success though - we prove upper and lower bounds which are within $d/2 + 1$ of each other for degree d, using a non-trivial reduction to results of Klawe [11] for pyramid graphs. In addition, we get tight results for height-3 trees and the height-4 binary tree.

THEOREM 2. *The fractional pebbling number of T_d^h is at least $(d-1)h/2 - d/2$, and at most $(d-1)h/2 + 1$.*

We then turn our attention to lower bounds. In our previous paper [3], we proved tight lower bounds for the tree evaluation problem on height-3 trees. Here we try to obtain lower bounds for trees of arbitrary height, but this comes at a cost to generality in the model. We introduce a natural semantic restriction on BPs which solve the tree evaluation problem: A k-way BP is *thrifty* if it only queries the function $f(x_1, \ldots, x_d)$ associated with a node when $(x_1, \ldots, x_d)$ are the correct values of the children of the node. The deterministic BPs corresponding to black pebbling are thrifty; so are the non-deterministic BPs corresponding to fractional pebbling.

THEOREM 3. *If p is the minimum number of pebbles required to black-pebble T_2^h then every deterministic thrifty BP solving $BT_2^h(k)$ (or $FT_2^h(k)$) requires $\Omega(k^p)$ states.*

For the decision problem $BT_2^h(k)$ there is indeed a non-thrifty deterministic BP improving on the bound by a factor of $\log k$, and this is tight for $h = 3$ [3]. But we have not been able to improve on thrifty BPs for solving any function problem $FT_d^h(k)$.

We have been able to prove that thrifty non-deterministic BPs cannot beat fractional pebbling for binary trees of height $h = 4$ or less, but for general trees this is open. It is not hard to see that for black pebbling, fractional pebbles do not help. The difficulty of analysing fractional pebbling compared to black pebbling might explain why we have been able to prove tight bounds for deterministic thrifty BPs for all binary trees, but only for trees of height 4 or less for non-deterministic thrifty BPs.

We pose the following as another interesting open question:

Thrifty Hypothesis: Thrifty BPs are optimal among k-way BPs solving $FT_d^h(k)$.

Proving this for deterministic BPs would show $\mathbf{L} \neq \mathbf{LogDCFL}$, and for non-deterministic BPs would show $\mathbf{NL} \neq \mathbf{LogCFL}$. Disproving this would provide interesting new space-efficient algorithms and might point the way to new approaches for proving lower bounds.

1.1 Relation to previous work

Taitslin [16] proposed a problem similar to $BT_2^h(k)$ in which the functions attached to internal nodes are specific quasi groups, in an unsuccessful attempt to prove $\mathbf{NL} \neq \mathbf{P}$.

Gal, Koucky and McKenzie [8] proved exponential lower bounds on the size of restricted n-way branching programs solving versions of the problem GEN. Like our problems $BT_d^h(k)$ and $FT_d^h(k)$, the best known upper bounds for solving GEN come from pebbling algorithms.

As a concrete approach to separating $\mathbf{NC}^1$ from $\mathbf{NC}^2$, Karchmer, Raz and Wigderson [10] suggested proving that the circuit depth required to compose a Boolean function with itself h times grows appreciably with h. Edmonds, Impagliazzo, Rudich and Sgall [7] noted that the approach would in fact separate $\mathbf{NC}^1$ from $\mathbf{AC}^1$. They also coined the name *Iterated Multiplexor* for the most general computational problem considered in [10], namely composing in a tree-like fashion a set of explicitly presented Boolean functions, one per tree node. Our problem $FT_d^h(k)$ can be considered as a generalisation of the Iterated Multiplexor problem in which the functions map $[k]^d$ to $[k]$ instead of $\{0,1\}^d$ to $\{0,1\}$. This generalisation allows us to focus on getting lower bounds as a function of k when the tree is fixed.

1.2 Organization

The paper is organized as follows. Section 2 defines the main notions used in this paper, including branching programs and pebbling. Section 3 proves various upper and lower bounds on black, black-white and fractional pebbling. Section 4 relates branching programs and pebbling, and uses the results of Section 3 to prove upper bounds on the size of branching programs. Section 5 contains results for thrifty branching programs. Because of space constraints, proofs are omitted from this version of the paper.

2 Preliminaries

We assume some familiarity with complexity theory, such as can be found in [9]. We write $[k]$ for $\{1, 2, \ldots, k\}$. For $d, h \geq 2$ we use T_d^h to denote the balanced d-ary tree of height h.

Warning: Here the *height* of a tree is the number of levels in the tree, as opposed to the distance from root to leaf. Thus T_2^2 has just 3 nodes.

We number the nodes of T_d^h as suggested by the heap data structure. Thus the root is node 1, and in general the children of node i are (when $d = 2$) nodes $2i, 2i + 1$ (see Figure 1).

DEFINITION 4. *[Tree evaluation problems] Given: The tree T_d^h with each non-leaf node i independently labelled with a function $f_i : [k]^d \to [k]$ and each leaf node independently labelled with an element from $[k]$, where $d, h, k \geq 2$.*
Function evaluation problem $FT_d^h(k)$: Compute the value $v_1 \in [k]$ of the root 1 of T_d^h, where in general $v_i = a$ if i is a leaf labelled a and $v_i = f_i(v_{j_1}, \ldots, v_{j_d})$ if the children of i are $j_1, \ldots, j_d$.

Boolean problem $BT^h_d(k)$: *Decide whether* $v_1 = 1$.

2.1 Branching programs

We use the following definition of branching programs, inspired by Wegener [17, p. 239] and by the k-way branching program model of Borodin and Cook [2].

DEFINITION 5. *[Branching programs] A* non-deterministic k-way branching program B *computing a total function* $g : [k]^m \to R$*, where* R *is a finite set, is a directed rooted multi-graph whose nodes are called* states*. Every edge has a label from* $[k]$*. Every state has a label from* $[m]$*, except* $|R|$ final *sink states consecutively labelled with the elements from* R*. An input* $(x_1, \ldots, x_m) \in [k]^m$ *activates, for each* $1 \leq j \leq m$*, every edge labelled* x_j *out of every state labelled* j*. A* computation *on input* $\vec{x} = (x_1, \ldots, x_m) \in [k]^m$ *is a directed path consisting of edges activated by* $\vec{x}$ *which begins with the unique start state (the root), and either it is infinite, or it ends in the final state labelled* $g(x_1, \ldots, x_m)$*, or it ends in a non-final state labelled* j *with no out-edge labelled* x_j *(in which case we say the computation* aborts*). At least one such computation must end in a final state. The* size *of* B *is its number of states.* B *is* deterministic k-way *if every non-final state has precisely* k *out-edges labelled* $1, \ldots, k$*.* B *is* binary *if* $k = 2$*.*

We say that B *solves a decision problem (relation) if it computes the characteristic function of the relation.*

A k-way branching program computing the function $FT^h_d(k)$ requires k^d k-ary arguments for each internal node i of T^h_d in order to specify the function f_i, together with one k-ary argument for each leaf. Thus in the notation of Definition 4, $FT^h_d(k)$: $[k]^m \to R$ where $R = [k]$ and $m = \frac{d^{h-1}-1}{d-1} \cdot k^d + d^{h-1}$. Also $BT^h_d(k)$: $[k]^m \to \{0,1\}$.

For fixed d, h we are interested in how the number of states required for a k-way branching program to compute $FT^h_d(k)$ and $BT^h_d(k)$ grows with k. We define $\#\mathsf{detFstates}^h_d(k)$ (resp. $\#\mathsf{ndetFstates}^h_d(k)$) to be the minimum number of states required for a deterministic (resp. non-deterministic) k-way branching program to solve $FT^h_d(k)$. Similarly we define $\#\mathsf{detBstates}^h_d(k)$ and $\#\mathsf{ndetBstates}^h_d(k)$ to be the number of states for solving $BT^h_d(k)$.

The next lemma shows that the function problem is not much harder to solve than the Boolean problem.

LEMMA 6. *[3]*

$$\#\mathsf{detBstates}^h_d(k) \leq \#\mathsf{detFstates}^h_d(k) \leq k \cdot \#\mathsf{detBstates}^h_d(k)$$
$$\#\mathsf{ndetBstates}^h_d(k) \leq \#\mathsf{ndetFstates}^h_d(k) \leq k \cdot \#\mathsf{ndetBstates}^h_d(k)$$

Next we introduce thrifty programs, a restricted form of k-way branching programs for solving tree evaluation problems. Thrifty programs efficiently simulate pebbling algorithms, and implement the best known upper bounds for $\#\mathsf{ndetBstates}^h_d(k)$ and $\#\mathsf{detFstates}^h_d(k)$, and are within a factor of $\log k$ of the best known for $\#\mathsf{detBstates}^h_d(k)$. In Section 4 we prove tight lower bounds for deterministic thrifty programs which solve $BT^h_d(k)$ and $FT^h_d(k)$.

DEFINITION 7. *[Thrifty branching program] A deterministic k-way branching program which solves $FT_d^h(k)$ or $BT_d^h(k)$ is* thrifty *if during the computation on any input every query $f_i(\vec{x})$ to an internal node i of T_d^h satisfies the condition that $\vec{x}$ is the tuple of correct values for the children of node i. A non-deterministic such program is* thrifty *if for every input every computation which ends in a final state satisfies the above restriction on queries.*

Note that the restriction in the above definition is semantic, rather than syntactic. It somewhat resembles the semantic restriction used to define incremental branching programs in [8]. However we are able to prove strong lower bounds using our semantic restriction, but in [8] a syntactic restriction was needed to prove lower bounds.

2.2 Pebbling

The pebbling game for dags was defined by Paterson and Hewitt [15] and was used as an abstraction for deterministic Turing machine space in [5]. Black-white pebbling was introduced in [6] as an abstraction of non-deterministic Turing machine space (see [14] for a recent survey).

Here we define and use three versions of the pebbling game. The first is a simple 'black pebbling' game: A black pebble can be placed on any leaf node, and in general if all children of a node i have pebbles, then one of the pebbles on the children can be slid to i (this is a "black sliding move')'. Any black pebble can be removed at any time. The goal is to pebble the root, using as few pebbles as possible. The second version is 'whole' black-white pebbling as defined in [6] with the restriction that we do not allow "white sliding moves". Thus if node i has a white pebble and each child of i has a pebble (either black or white) then the white pebble can be removed. (A white sliding move would apply if one of the children had no pebble, and the white pebble on i was slid to the empty child. We do not allow this.) A white pebble can be placed on any node at any time. The goal is to start and end with no pebbles, but to have a black pebble on the root at some time.

The third is a new game called *fractional pebbling*, which generalises whole black-white pebbling by allowing the black and white pebble value of a node to be any real number between 0 and 1. However the total pebble value of each child of a node i must be 1 before the black value of i is increased or the white value of i is decreased. Figure 2 illustrates two configurations in an optimal fractional pebbling of the binary tree of height three using 2.5 pebbles.

Our motivation for choosing these definitions is that we want pebbling algorithms for trees to closely correspond to k-way branching program algorithms for the tree evaluation problem.

We start by defining fractional pebbling, and then define the other two notions as restrictions on fractional pebbling.

DEFINITION 8. *[Pebbling] A* fractional pebble configuration *on a rooted d-ary tree T is an assignment of a pair of real numbers $(b(i), w(i))$ to each node i of the tree, where*

$$0 \leq b(i), w(i) \tag{1}$$

$$b(i) + w(i) \leq 1 \tag{2}$$

Here $b(i)$ and $w(i)$ are the black pebble value *and the* white pebble value*, respectively, of i, and $b(i) + w(i)$ is the* pebble value *of i. The number of pebbles in the configuration is the sum over all nodes i of the pebble value of i. The legal pebble moves are as follows (always subject to maintaining the constraints (1), (2)): (i) For any node i, decrease $b(i)$ arbitrarily, (ii) For any node i, increase $w(i)$ so that $b(i) + w(i) = 1$, (iii) For every node i, if each child of i has pebble value 1, then decrease $w(i)$ to 0, increase $b(i)$ arbitrarily, and simultaneously decrease the black pebble values the children of i arbitrarily.*

A fractional pebbling *of T using p pebbles is any sequence of (fractional) pebbling moves on nodes of T which starts and ends with every node having pebble value 0, and at some point the root has black pebble value 1, and no configuration has more than p pebbles.*

A whole black-white pebbling *of T is a fractional pebbling of T such that $b(i)$ and $w(i)$ take values in $\{0,1\}$ for every node i and every configuration. A* black pebbling *is a black-white pebbling in which $w(i)$ is always 0.*

Notice that rule (iii) does not quite treat black and white pebbles dually, since the pebble values of the children must each be 1 before any decrease of $w(i)$ is allowed. A true dual move would allow increasing the white pebble values of the children so they all have pebble value 1 while simultaneously decreasing $w(i)$. In other words, we allow black sliding moves, but disallow white sliding moves. The reason for this (as mentioned above) is that non-deterministic branching programs can simulate the former, but not the latter.

We use $\#\mathsf{pebbles}(T)$, $\#\mathsf{BWpebbles}(T)$, and $\#\mathsf{FRpebbles}(T)$ respectively to denote the minimum number of pebbles required to black pebble T, black-white pebble T, and fractional pebble T. Bounds for these values are given in Section 3. For example for $d = 2$ we have $\#\mathsf{pebbles}(T_2^h) = h$, $\#\mathsf{BWpebbles}(T_2^h) = \lceil h/2 \rceil + 1$, and $\#\mathsf{FRpebbles}(T_2^h) \leq h/2 + 1$. In particular $\#\mathsf{FRpebbles}(T_2^3) = 2.5$ (see Figure 2).

3 Pebbling Bounds

3.1 Previous results

We start by summarizing what is known about whole black and black-white pebbling numbers as defined at the end of Definition 8 (i.e. we allow black sliding moves but not white sliding moves).

The following are minor adaptations of results and techniques that have been known since work of Loui, Meyer auf der Heide and Lengauer-Tarjan [13, 1, 12] in the late '70s. They considered pebbling games where sliding moves were either disallowed or permitted for both black and white pebble, in contrast to our results below.

We always assume $h \geq 2$ and $d \geq 2$.

THEOREM 9. *[3]* $\#\mathsf{pebbles}(T_d^h) = (d-1)h - d + 2$.

THEOREM 10. *For $d = 2$ and d odd:*

$$\#\mathsf{BWpebbles}(T_d^h) = \lceil (d-1)h/2 \rceil + 1 \tag{3}$$

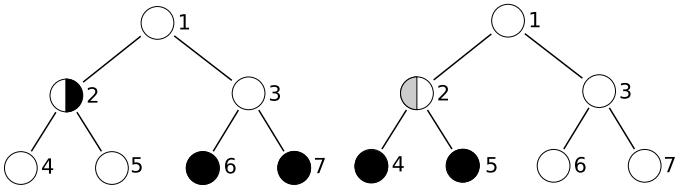

Figure 2: Two configurations from the pebbling of the height 3 binary tree with 2.5 pebbles

For d even:

$$\#\mathsf{BWpebbles}(T_d^h) \leq \lceil (d-1)h/2 \rceil + 1 \tag{4}$$

When d is odd, this number is the same as when white sliding moves are allowed.

3.2 Results for fractional pebbling

The concept of fractional pebbling is new. Determining the minimum number p of pebbles required to fractionally pebble T_d^h is important since $O(k^p)$ is the best known upper bound on the number of states required by a non-deterministic BP to solve $FT_d^h(k)$ (see Theorem 18). It turns out that proving fractional pebbling lower bounds is much more difficult than proving whole black-white pebbling lower bounds. We are able to get exact fractional pebbling numbers for the binary tree of height 4 and less, but the best general lower bound comes from a nontrivial reduction to a paper by Klawe [11] which proves bounds for the pyramid graph. This bound is within $d/2 + 1$ pebbles of optimal for degree d trees (at most 2 pebbles from optimal for binary trees).

Our proof of the exact value of $\#\mathsf{FRpebbles}(T_2^4) = 3$ led us to conjecture that any non-deterministic BP computing $BT_2^4(k)$ requires $\Omega(k^3)$ states. In Section 5 we provide evidence for that conjecture by proving that any non-deterministic *thrifty* BP requires $\Omega(k^3)$ states.

We start by presenting a general result showing that fractional pebbling can save at most a factor of two over whole black-white pebbling for any DAG (directed acyclic graph). (Here the pebbling rules for a DAG are the same as for a tree, where we require that every sink node (i.e. every 'root') must have a whole black pebble at some point.) We will not use this result, but it does provide a simple proof of weaker lower bounds than those given in Theorem 12 below.

THEOREM 11. *If a DAG D has a fractional pebbling using p pebbles, then it has a black-white pebbling using $2p$ pebbles.*

The next result presents our best-known bounds for fractionally pebbling trees T_d^h. Theorem 2 is the first part of this result.

THEOREM 12.

$$(d-1)h/2 - d/2 \leq \#\mathsf{FRpebbles}(T_d^h) \leq (d-1)h/2 + 1$$

$$\#\mathsf{FRpebbles}(T_d^3) = (3/2)d - 1/2$$

$$\#\mathsf{FRpebbles}(T_2^4) = 3$$

Theorem 12 is a consequence of the following lemmas.

LEMMA 13.

$$\#\mathsf{FRpebbles}(T_d^h) \leq (d-1)h/2 + 1$$

This lemma gives the upper bound for all degrees and heights.

LEMMA 14.

$$\#\mathsf{FRpebbles}(T_d^3) = (3/2)d - 1/2$$

This lemma gives the lower bound for height 3 and all degrees. It follows from the asymptotically tight lower bound on the number of states for non-deterministic BPs computing $BT_d^3(k)$ in [3] (Theorem 4.3 in that paper).

LEMMA 15.

$$\#\mathsf{FRpebbles}(T_2^4) \geq 3$$

This lemma gives the tight lower bound for binary height 4 trees.

LEMMA 16. *For any d and h,* $\#\mathsf{FRpebbles}(T_d^h \geq (d-1)(h-1)/2 - .5$

This lemma gives our general lower bound for all degrees and heights. We do not believe that this lower bound is tight. The proof of Lemma 16 requires the following result about optimal pebblings.

LEMMA 17. *For every finite DAG there is an optimal fractional B/W pebbling in which all pebble values are rational numbers. (This result is robust independent of various definitions of pebbling; for example with or without sliding moves, and whether or not we require the root to end up pebbled.)*

4 Pebbling and Branching Program Upper Bounds

In this section, we connect pebbling upper bounds with upper bounds for branching programs, and use the results of the previous section to derive tight bounds for branching program size of tree evaluation on trees of small height.

The first result connects pebbling upper bounds with upper bounds for thrifty branching programs. The second part is Theorem 1. Part (i) of this result without the thriftiness condition was proved in [3].

THEOREM 18. *(i) If T_d^h can be black pebbled with p pebbles, then deterministic thrifty branching programs with $O(k^p)$ states can solve $FT_d^h(k)$ and $BT_d^h(k)$. (ii) If T_d^h can be fractionally pebbled with p pebbles then non-deterministic thrifty branching programs can solve $BT_d^h(k)$ with $O(k^p)$ states.*

COROLLARY 19. $\#\mathsf{ndetFstates}_d^h(k) = O(k^{\#\mathsf{FRpebbles}(T_d^h)})$.

For every height $h \geq 2$ we prove upper bounds for deterministic *thrifty* programs which solve $FT_d^h(k)$ (Theorem 20, (5)), and show in Section 5 that these bounds are optimal for degree $d = 2$ even for the Boolean problem $BT_d^h(k)$ (Theorem 21). We prove upper bounds for non-deterministic thrifty programs solving $BT_d^h(k)$ in general, and show in Section 5 that these are optimal for binary trees of height 4 or less (Theorem 22 together with Theorem 4.3 in [3]).

For the non-deterministic case our best BP upper bounds for every $h \geq 2$ come from fractional pebbling algorithms via Theorem 18. For the deterministic case our best bounds for the function problem $FT_d^h(k)$ come from black pebbling via the same theorem, although we can improve on them for the Boolean problem $BT_2^h(k)$ by a factor of $\log k$ (for $h \geq 3$) [3].

THEOREM 20. *[BP Upper Bounds] For all $h, d \geq 2$*

$$\#\mathsf{detFstates}_d^h(k) = O(k^{(d-1)h-d+2}) \quad (5)$$

$$\#\mathsf{ndetBstates}_d^h(k) = O(k^{(d-1)(h/2)+1}) \quad (6)$$

These bounds are realized by thrifty programs.

5 Thrifty lower bounds

See Definition 7 for thrifty programs.

Theorem 21 below shows that the upper bound given in Theorem 20 (5) is optimal for deterministic thrifty programs solving the function problem $FT_d^h(k)$ for $d = 2$ and all $h \geq 2$. Theorem 22 shows that the upper bound given in Theorem 20 (6) is optimal for non-deterministic thrifty programs solving the Boolean problem $BT_d^h(k)$ for $d = 2$ and $h = 4$. Theorem 21 below is a re-statement of Theorem 3.

THEOREM 21. *For all $h \geq 2$ every deterministic thrifty program that solves $BT_2^h(k)$ has at least $0.5k^h$ states for sufficiently large k.*

Next we prove a lower bound on non-deterministic thrifty branching programs.

THEOREM 22. *Every non-deterministic thrifty branching program solving $BT_2^4(k)$ has $\Omega(k^3)$ states.*

6 Conclusion

The Thrifty Hypothesis states that thrifty branching programs are optimal among k-way BPs solving $FT_d^h(k)$. For the deterministic case, this says that the black pebbling method is optimal. Proving this would separate **L** from **P** .

Even disproving this would be interesting, since it would show that one can improve upon this obvious application of pebbling. One of the referees pointed out that if the functions at nodes are restricted to be integer polynomials, then for some parameter settings it is possible to obtain non-trivial branching programs that are not thrifty, by using Chinese remaindering.

Other accessible open problems are to generalise Theorem 22 to get general lower bounds for non-deterministic thrifty BPs solving BT_2^h, and to improve Theorem 12 to get tight bounds on the number of pebbles required to fractionally pebble T_d^h.

For a complete and combined treatment of the notions and results in this paper and in [3], please see [4].

Acknowledgement James Cook played a helpful role in the early parts of this research.

References

[1] F. Meyer auf der Heide. A comparison between two variations of a pebble game on graphs. Master's thesis, Universität Bielefeld, Fakultät für Mathematik, 1979.

[2] A. Borodin and S. Cook. A time-space tradeoff for sorting on a general sequential model of computation. *SIAM J. Comput.*, 11(2):287–297, 1982.

[3] M. Braverman, S. Cook, P. McKenzie, R. Santhanam, and D. Wehr. Branching programs for tree evaluation. In *Proceedings of 34th International Symposium on Mathematical Foundations of Computer Science*, 2009.

[4] M. Braverman, S. Cook, P. McKenzie, R. Santhanam, and D. Wehr. Pebbles and branching programs for tree evaluation. On Steve's webpage, 2009.

[5] S. Cook. An observation on time-storage trade off. *J. Comput. Syst. Sci.*, 9(3):308–316, 1974.

[6] S. Cook and R. Sethi. Storage requirements for deterministic polynomial time recognizable languages. *J. Comput. Syst. Sci.*, 13(1):25–37, 1976.

[7] J. Edmonds, R. Impagliazzo, S. Rudich, and J. Sgall. Communication complexity towards lower bounds on circuit depth. *Computational Complexity*, 10(3):210–246, 2001. An abstract appeared in the *32nd IEEE FOCS (1991)*.

[8] A. Gál, M. Koucký, and P. McKenzie. Incremental branching programs. *Theory Comput. Syst.*, 43(2):159–184, 2008.

[9] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[10] M. Karchmer, R. Raz, and A. Wigderson. Super-logarithmic depth lower bounds via direct sum in communication complexity. *Computational Complexity*, 5:191–204, 1995. An abstract appeared in the *6th Structure in Complexity Theory Conference (1991)*.

[11] M. Klawe. A tight bound for black and white pebbles on the pyramid. *J. ACM*, 32(1):218–228, 1985.

[12] T. Lengauer and R. Tarjan. The space complexity of pebble games on trees. *Inf. Process. Lett.*, 10(4/5):184–188, 1980.

[13] M.C. Loui. The space complexity of two pebble games on trees. Technical Report LCS 133, MIT, Cambridge, Massachussetts, 1979.

[14] J. Nordström. New wine into old wineskins: A survey of some pebbling classics with supplemental results. Available on line at `http://people.csail.mit.edu/jakobn/research/`, 2009.

[15] M. Paterson and C. Hewitt. Comparative schematology. In *Record of Project MAC Conference on Concurrent Systems and Parallel Computations*, pages 119–128, 1970. (June 1970) ACM. New Jersey.

[16] M.A. Taitslin. An example of a problem from PTIME and not in NLogSpace. In *Proceedings of Tver State University*, volume 6(12) of *Applied Mathematics, issue 2, Tver State University, Tver*, pages 5–22, 2005.

[17] I. Wegener. *Branching Programs and Binary Decision Diagrams*. SIAM Monographs on Discrete Mathematics and Applications. Soc. for Industrial and Applied Mathematics, Philadelphia, 2000.